(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 364 849 A2**

(12) **EUROPEAN PATENT APPLICATION**

(21) Application number: **89118714.8**

(51) Int. Cl.5: **G06F 13/42**

(22) Date of filing: **09.10.89**

(30) Priority: **21.10.88 US 261047**

(43) Date of publication of application: **25.04.90 Bulletin 90/17**

(84) Designated Contracting States: **AT BE CH DE ES FR GB GR IT LI LU NL SE**

(71) Applicant: **Siemens Aktiengesellschaft**
Wittelsbacherplatz 2
D-8000 München 2(DE)

(72) Inventor: **Johnson, David**
3401 SW Stonebrock
Portland Oregon 97201(US)
Inventor: **Myers, Mark**
7025 SW Burlingame
Portland Oregon 97219(US)
Inventor: **Ebersole, Ronald**
145 SW Kennedy
Beaverton Oregon 97005(US)
Inventor: **Budde, David**
2825 SW 90
Hillsboro Oregon 97123(US)
Inventor: **Bier, Gerhard**
Blumenstrasse 1
D-6742 Herxheim-Hayna(DE)

(54) Bus for a data processing system.

(57) An advanced processor bus system communicates data words among processors connected to the advanced processor bus by means of bus expander interface units. Memory arrays are connected to the advanced processor bus by means of memory control interface units. The interface units issue data requests for the transmission of data upon the advanced processor bus for reception into a register of the requesting interface units. The interface units then issue data replies which are communicated in packets by message types over the advanced processor bus in response to the data requests, with each packet occupying at least one time slot. Each time slot is capable of including operation specification code, address, data, control, and parity-check bits. Messages are qued and controlled for pipelining. The action on the bus can be frozen for one cycle repetitively for diagnosis purposes.

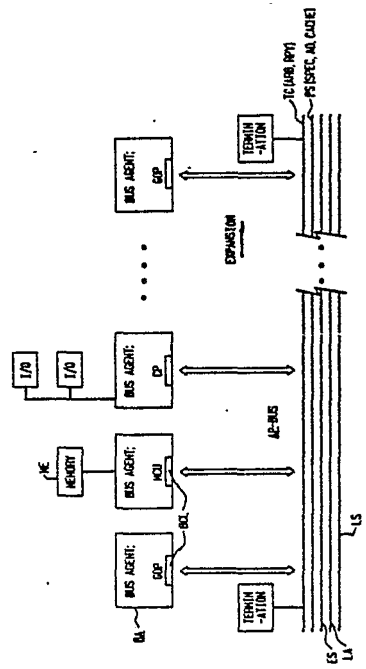


FIG. 1

BUS FOR A DATA PROCESSING SYSTEM

Field of the Invention

This invention relates to electronic data processing systems and more particularly to the method and apparatus of an advanced processor bus system, communicating between and interconnecting with the major branches of the data processing system.

Background of the Invention

In the design of a data processing system, one of the first areas of design allocation is the data communication capability. Since, except for very simple designs, every component cannot be directly connected to every other component with a full compliment of address and data lines, as even Very Large Scale Integrated (VLSI) circuits have pin out and drive limitations, the necessity of a system bus interconnecting the data processing system providing time division multiplexed sharing of the address and data portions of the bus between communicating components, becomes obvious. Once time division multiplexing of address and data information on a bus is required, so are control signals, timing signals, defined operations and protocols in order to operate the bus such that it uses the available data-bandwidth efficiently.

In order to achieve a high data bandwidth, a system bus usually is designed with numerous parallel lines so a unit of data referred to as a "word" can be transmitted at one time, and it is fashioned to transfer these "words" as quickly as feasible (i.e., it operates at high clock rates). To use the high data bandwidth effectively, control lines are included in parallel to the data lines, and a structure of bus sequences is defined so information can be communicated whenever possible and therefore the bus is never left waiting while there are available operations to be performed.

Another attribute of an effective bus system is that data scheduled to be transmitted over the bus system is provided the opportunity to reach its destination with a limited number of delays. The delay from the time the data is available to be transmitted on the bus, until the time it arrives at its destined address in clock cycles, is defined as the latency. Some latency is the inherent price paid for ordered communications at a high effective data bandwidth; however, the problems of data becoming invalid because of intervening updates while waiting for the bus at Input/Output connections, and at memory locations, require latency be minimized and defined always as a system operational limitation.

A further attribute of an effective bus system is that it must be flexible, that is, capable of communicating information to a variety of component configurations. It must be able to communicate to the bus interface unit of a processor, and the memory interface unit or units to form a single processor data processing system, and also capable of communicating to the bus interfaces of multiple general data processors, multiple memory interface units, and to the bus interface unit of input/output multiple processors to form multiprocessor, multibus systems. Flexibility comes by having a bus system that is modular and expandable, not only in hardware, but also modular and expandable in communication and data structure.

One example of a data processing bus system is shown in Budde et al. U. S. Patent No. 4,480,307 of October 30, 1984. Although a good bus system, it has a lower effective data bandwidth, primarily because it has only 16 data lines over which to send 32-bit data words and therefore takes at least two clock cycles per 32-bit word instead of one. Also, data going from a data processor unit to a memory control unit (MCU) has to go through a processor memory bus interface unit (BIU) where the communication is examined, the information format charged, and then switched onto a different format memory bus. This examination, change of format and switching time increases the latency of the data going between the general data processor and the interface memory controller. Moreover, the system bus provided for in the patent by Budde et al is somewhat inflexible. There is no mechanism to provide for data that takes a long time to access, such as information calculated by another processor, or stored in a slowly responsive memory. Data must be returned in the same order requested which requires either ensure fast responses or have the danger of waiting a long time in the FIFO response pipeline.

Thus, it is an object of this invention to provide a data processing bus system that has a high effective data bandwidth. A second object of this invention is to provide a data processing bus system which has low data latency. A further object of this invention is to provide a flexible data processing bus system that is capable of being expanded to systems with multiprocessors and multi-buses.

Summary of the Invention

Briefly stated in accordance with one aspect of the invention, the aforementioned objects are achieved by providing an Advanced Processor Bus system, an Advanced Processor Bus system for communicating data words among at least one processor connecting to the AP Bus by at least one bus interface means such as a Bus Expander Unit; and at least one memory array connecting to the AP Bus by at least memory control mans, such as a Memory Control unit, wherein said bus interface means connected to respective processors issue data requests request the transmission of data upon the AP Bus for reception into a register of each said bus interface means making such request; and said bus interface means and said memory control means issue data replies communicating over the AP Bus in response to the data requests by transmission of the requested data. The AP Bus system comprises a message generator means for generating messages in the form of packets of information for transmission on said AP Bus system, said messages being divided into message types including control message packets, request message packets, and reply message packets, each packet occupying at least one bus transmission time slot issued by said message generator means sequentially and contiguously, each bus transmission time slot in a packet being capable of including operation specification code, address, data, control, and parity-check bits; a pipeline queue; and message controller means connected to said message generator means and to said pipeline queue for controlling said request message packets, said control message packets and said reply message packets such that a predetermined number of said request message packets and control message packets requiring reply message packets may be entered into said pipeline queue at any one time.

A monitor means is connected to said message controller means and to said AP Bus system for monitoring said request message packets and control message packets, and said reply message packets communicated on said AP Bus by said message generator, preventing request message packet replies in excess of said predetermined number from being communicated until a reply message packet is communicated to thereby make availble a slot in the pipeline queue.

Control signal connections, as a part of said AP Bus operative in parallel with the data portion thereof, provide a coded signal indicating a particular message type communicated on said bus by said message generator.

An interface logic means is connected to said message generator means and to said control signal lines for communicating over said AP Bus system the type of message packet generated.

Said message generator means includes means for inserting a particular reply message packet corresponding to a particular request message packet in said pipeline queue at a position in said pipeline queue corresponding to the request message packet that is associated with said particular reply message packet.

In another aspect of the invention, the objects are achieved by providing an Advanced Processor Bus system for use in a data processing system using VLSI Advanced Processor components, having AP Bus interface units for processors and memory interface units as AP Bus agents, said AP Bus connecting to and communicating among each of said agents, comprising recognizing means connected to said Advanced Processor Bus for recognizing signals on said AP Bus representative of a request to one of said agents.

A means is connected to said AP Bus, responsive to said recognizing means, for generating on said Bus, signals representative of a first and a second write-request packets, and a first and second read request packets, said packets comprising a number of bus transmission time slots, issued sequentially and contiguously, each slot in a packet including operation-specification bits; address bits; data bits; and control bits. Said first write-request packet includes an operation specification code specifying a write function to be performed and a memory location where the information bits are written to, and said second write-request packet including an operation specification code specifying a function to be performed and a memory mapped location where the information bits are to be written to. Said first read request packet includes an operation specification code specifying a read function to be performed and a memory location where the information bits are read from, and said second read request packet including an operation specification code specifying a function to be performed and a memory mapped location where the information bits are read from. Said memory mapped locations correspond to inter-agent communications specifying an interval destination code of at least one of said agents and the functions desired of the at least one agent.

A receiving means is connected to said AP Bus for receiving a reply message packed on said AP Bus System, said reply message packet including a number of bus transmission time slots, received sequentially and contiguously, each said time slot in said reply packet including operation specification code, data bits and control bits.

Said reply message packet contains an operation specification code and if requested data.

A message control means is connected to said generating means and said receiving means said message control means including as a part of said message control means: a means for queuing data in said write request packets; a means responsive to said control bits in said read packets, and write packets for detecting a beginning and an end of each packet; a means for decoding each said operation specification code to indicate the operation specified; a means for establishing a pipeline for said request message packets, and said reply message packets generated on said AP Bus by said message generator; a monitor means connected to said message control means and to said AP Bus system, for monitoring signals on said AP Bus system representative of said request packets, and reply packets on said bus, such that a request packet is prevented from entering said pipeline until a reply packet is received to make available a slot in said pipeline. Request packets increase the length of said pipeline up to a predetermined number of request packets in said pipeline, and reply packets decrease the length of said pipeline. A means is connected to said recognizing means, said generating means, said message control means and said receiving means, for arbitrating said signals on said Bus system representative of requests going onto said AP Bus system and said signals on said AP Bus system representative of replies received on said AP Bus, enabling said message control means to insert a reply packet associated with a particuluar request packet in the pipeline position of said recognized request packet.

In both aspects of the AP Bus system, there is a command for use in diagnostics called the WAIT command which when given freezes all action on the bus for one cycle, and repetitively issuing the WAIT command freezes the AP Bus totally.

Brief Description of the Drawings

While the specification concludes with claims particularly pointing out and distinctly claiming the subject matter which is regarded as the invention, it is believed that the invention will be better understood from the following description of the preferred embodiment taken in conjunction with the accompanying drawings in which:

Figure 1 shows a simplified partial block diagram of an AP Bus system among agents;

Figure 2 shows a functional block diagram of a BXU agent illustrating the function of the signal lines of the AP Bus and the Local bus;

Figure 3 shows a timing diagram of various AP Bus operations;

Figure 4 shows the specification field of a packet and the significance of the signals;

Figure 5 shows a timing diagram of how the next available bus cycle is determined.

Figure 6 shows in block diagram form the ordering of Request Packets and Reply Packets;

Figure 7 shows the two fields that make up the ARBID code; and

Figure 8 shows a timing diagram of a four agent arbitration and access.

TABLE OF CONTENTS

A. AP-Bus Fundamentals:

1. Introduction:

The AP Bus is a 32-bit, synchronous, bi-directional bus with multiplexed address and data. Any agent, being defined as any BXU or MCU on the AP Bus that meets the AP Bus specification, may communicate with another agent including memory via a bus transaction. A bus transaction consists of two communication packets, a Request packet sent by the node associated with the agent initiating the transaction and the Reply packet from the replying agent that completes it. A packet is a block of information that is logically connected and transmitted in sequence on the bus. Packets may be from one to five bus cycles in length and contain from one to sixteen bytes of data. Address information is always contained in the first cycle of any Request packet. Control information is included with each request or reply packet cycle. The address contains a memory address for a memory transaction or the memory-mapped address of another agent for an interprocessor transaction to the processor connected to the other agent on a local bus.

The AP Bus provides a high performance communication system for both single and multi-processor configurations with either single or multiple AP Bus topologies. The AP Bus system allows several Request Packets to concurrently wait for their respective replies. The mechanism that maintains the ordering of the replies and facilitates the completion of the transactions is referred to as the Pipeline Queue.

Requests are scheduled in a Grant Queue which holds the results of a sequential parallel polling process. In each step of the sequence, a group of bus agents will be polled if any want access to the bus. An arbitration mechanism provides quick resolution between conflicting requests.

The arbitration and control functions are distributed across the agents attached to the bus. There are no centralized arbitration or control components.

2. Structure:

Several bus agents are shown with their respective attached branches to an AP Bus in Figure 1. The GDP general data processor can communicate, via its respective Bus Expander Unit, with the memory array MA via a memory controller unit, MCU or a Bus Expander Unit or another processor via a BXU. The MCU or BXU provides the logic for interfacing both to the bus and to the memory MA. It will convert a bus transaction to an access or series of accesses to the memory array. The AP Bus system also provides the medium and protocol by which processors communicate between themselves. Figure 1 illustrates how additional processors and memory arrays and their associated BXU or MCU may be added to the bus to expand its memory, processing power, or functionality. Figure 1 also illustrates how complex single bus topologies can be connected through the use of bus expander units.

The AP Bus handles 47 bi-directional signals and 8 clock and control signals. All BXU's and MCU's attach to the 47 bi-directional bus lines. The 8 clock and control lines are either sourced to all agents or are local signals that are used for bus-related functions. The bi-directional bus signals are 'wire-OR'ed. This means that more than one MCU or BXU may assert a control or data signal at the same time. The signal will have the same value if one or more agents assert it. The asserted state is a low voltage level on the physical bus. The non-asserted or idle state will be high voltage.

3. AP-Bus Signals:

3.1. Transaction Control Signal Group, TC (5 total)

These signals as shown in Figures 1 and 2, consist of the arbitration signals and the Reply Ordering signal.

3.1.1. Arbitration: lines ARB (3..∅)

The ARB signals as shown in Figure 8 are used by the bus agents to determine which agent has access to the bus next.

3.1.2. Reply Ordering: line RPY DEF

The Reply Defer signal allows an agent to give up its "time slot" on the bus if its access is going to take a long time.

3.2. Packet Signals, PS (38 total)

Packet Signals and signal lines as shown in Figures 2 and 3 are bi-directional and collectively form the actual packet. Address, data, and the type of transaction are transmitted on these lines.

3.2.1. Packet Specification: lines SPEC (5..0)

The signals transmitted over the SPEC lines define the packet type, i.e., operation type, and the parameters required for the transaction.

3.2.2. Address/Data: lines AD (31..∅)

The AD lines transmit address and data information in a time progression during the transaction. The content of the AD lines is defined by the SPEC encoding during the same initial bus cycle.

3.3. Error Signal Group (lines ES) (4 total)

These signals eiher provide redundancy to allow error detection and/or signal errors to other bus agents. They are part of a complete fault tolerant support package and represent the bus level portion of the support. They may be used independently of the higher levels of support.

3.3.1. Check Signal: CHK (1, ∅)

Provide parity for SPEC lines and for AD lines signals.

3.3.2. Bus Error Signals: BERL (1, ∅)

BERL is used to signal errors from bus operations or within BXU's/MCU's and associated circuitry.

3.4. Synchronization and Initialization Signal Group (lines LS) (3 total)

These signals provide the ability to bring all bus agents to a consistent state and control the timing of bus signals. They are: SYSTEM CLOCK - 2XCLK; INITIALIZATION Signal - INIT; and System Debug and Test - WAIT.

3.5. Local Agent Signal Group (lines LA) (5 total)

The local agent signals have meaning only for a single agent but are commonly used signals. The system clock frequency, 2XCLK, is distributed to all Bus agents. This clock frequency will be divided in half to obtain the frequency at which the bus will transfer information. This clock period will be referred to as a bus cycle BC or just cycle throughout the description. The bus cycle provides the reference for all signals. Most bus signals will be driven in the beginning of the bus cycle and sampled at the end, the exceptions being the arbitration and the BERL signals which are driven midway from the beginning of a bus clock cycle. Figure 3 illustrates the basic relationship between clocks and bus signals.

The SYSTEM CLOCK CLK shown in Figure 3 is the clock distributed to all bus agents. The bus cycle is shown as two system clock periods in length and each one is given a number M. The bus cycle will be shown in all timing diagrams while the system clock may be omitted for clarity. The falling edge is shown as coincident with the division line while the bus data BD is shown as slightly offset. This illustrates that the falling clock edge is used to sample the data at the end of the bus cycle. Bus data is shown as being driven after the falling edge of the clock.

Figure 5 illustrates how the bus state is sampled and used to determine the next data to be driven on the bus. All agents keep track of what can happen next on the bus and then modify it by information from the cycle in progress.

3.6.

The signal group provides the basis for the operation of the various bus agents in a system environment. This signal group supports the functions necessary for the power-up and initialization of an on-line replaced circuit board. The signal group provides communication to the additional system internal unit such as the power supply modules.

B. Bus Transactions:

An agent attached to the AP Bus will communicate with AP Memory, MA or another agent by a transaction. The transaction communicates the operation to be performed, the location where it will be performed and the amount of data involved.

Transactions are separated into two independent parts, a Request Packet and a Reply Packet. A packet is a sequential group of cycles on the bus lines that form a logical unit. The SPEC and AD lines during each cycle of the packet will convey operation specification, address, or data information for the packet. Operation specification information is transferred on the SPEC lines, while address and data are transferred on the AD (address and data) lines. The packet is first categorized into the Request or Reply type. If it is a request, it is broken down into a basic action which defines whether data is being transferred to (Read) or from (Write) the initiating agent. The Reply will indicate the completion status of requested operation with either an accepted or refusal Reply.

A Request is further divided into specific operations of the two basic actions. The specific operations determine the amount of data to be transferred and special variants. The Replies are also divided into specific results. The specific result can indicate that the Request was accepted or rejected, and the amount of data that is included in the Reply Packet.

The location of a bus memory transaction is defined by a 32-bit memory address. Each address points at a single bye within a large 16-byte block. All transactions will perform an operation on a block or a portion of a block. Memory transactions can access only one block at a time. Advanced Processor Memory consists of elements that may be written to and read from using a specific set of operations. An agent can utilize AP Memory for both instruction and operand storage.

AP Memory is logically divided into 16-byte blocks. All accesses to memory over the AP Bus are made to a block. One to sixteen bytes of data may be read from or written to a block in any single transaction. Blocks are defined to begin on every 16-byte boundary beginning at address 0000 0000(hex). When the low-order four bits of an address are zero then the address points at the first byte in a block.

Each block is divided into bytes and words. A byte is eight bits long and is the minimum unit of memory that can be accessed by any transaction. All memory addresses point at a byte in memory. There are four bytes in a word and four words in a block. A memory address with the low order two bits equal to zero will point at the first byte (byte 0) in a word. The word is the basic unit of transfer on the Address/Data lines of the bus.

A transaction can access only a single block. Any agent desiring to access more than 16 bytes or data that crosses a block boundary must break those accesses into multiple transactions. In addition, the transaction to a block must explicitly access a contiguous string of data within the block. Access to the second and fourth bytes within a block cannot be done in one transaction as they are not contiguous.

The location of a bus memory-mapped transaction is likewise defined by a 32-bit memory-mapped address. Memory-mapping can use the accessed address space for other than memory operations. The initialization commands used to reset and parameterize bus agents are an example of this. These "commands" can only be written and will be completed differently than if the same transaction were directed at AP Memory. A memory-mapped space may use a smaller subset of the defined operations, implement a function other than memory, or complete the operation with a different Reply than AP Memory would. Both AP Memory and memory-mapped transactions will look like similar transactions when viewed on the bus, but will produce different actions for the same fundamental operations. Inter-Agent Communication (IAC) is a specific set of memory-mapped addresses recognized by all agents, i.e., the BXU's and MCU's. IAC's provide a non-memory based, communication between bus agents and their attached processors or memory arrays. They are used for such system functions as initialization, access to error logs, and interrupt handling at the overall system level.

The 32-bit address field on the AP Bus provides a 4 Gigabyte address space. The top 12 Megabytes are reserved for IACs. The next 4 Megabytes are reserved for ICE memory. The remaining space is available for normal memory and memory mapped devices. AP Memory and IAC transactions will be defined more explicitly below.

A Request Packet will transfer data for the Write operation and the Reply Packet for the Read operation. Both packets will transfer up to 16 bytes of data on the AD lines, taking from 1 to 4 cycles in the packet depending on the amount of data. Two cycles is the implemented minimum number of cycles for a packet; if only one cycle is required for data transfer it is followed by a null cycle. The data is organized as words within the block and will be transferred as a word on the AD lines.

Each byte position in a memory word will always occupy the same position on the AD lines. Byte 0 in a word will be transferred on AD0 thru AD7, byte 1 on AD8 thru AD15, and so forth. These positions on the AD lines will also be referred to as byte 0, byte 1, etc. Since data is not justified on the bus based on address, the requesting agent must reorder the data as required.

The SPEC field within a cycle is used to transfer the Tag Bit for both Reads and Writes. The Tag Bit will be the one associated with the word being transferred on the AD lines during the same cycle.

C. Read Data Transfer:

The word is the basic unit of transfer on the bus for a Read transaction. A full bus cycle is required to transfer a word or part of the word. The requester must, therefore, formulate a Request for the words that include the desired string of data in the block. The address in the Request will determine the first word to be transferred. The initiating agent will receive the word(s) and extract the bytes desired, discarding those not needed.

D. Write Data Transfer:

The Write Request is also word oriented in the transfer of data on the AD lines. Unlike reads, where data is not modified by the transaction, the Write Request must define the locations to be altered. The Write Request uses Byte Marks to define the specific bytes within a word that will be written. There is a Byte Mark for each byte of data on the AD lines transferred on the SPEC lines during the same cycle. When the Byte Mark is asserted, the byte on the AD lines will be written.

Write data will be transferred with the Write Request Packet. The first cycle will define the specific operation to be performed and transmit the address at which it will be performed. The Write data will follow the first cycle. The SPEC lines for each word will contain both the Tag Bit and the Byte Marks. The address will point at the first byte to be written in the block being accessed. A contiguous string of data will be written starting at that byte. Its length is defined by the number of cycles and the Byte Marks in each cycle.

E. IAC Transactions:

The AP Bus defines a set of memory-mapped primitives to perform system initialization. These are called Inter-Agent Communication or IACs. The uppermost 12 megabytes of the AP Bus address space is dedicated to IACs and redefined to provide agent addressing and IAC parameters.

The memory-mapped address is broken down into IAC access type, bus destination and internal destination. The type field identifies which of the three agent addressing methods will be used.

These spaces are for LOGICAL ID, PHYSICAL ID and presently unused space. The bus destination field contains the ID defined by the access type and will be used to address the intended agent. The LOGICAL and PHYSICAL IDs are used for normal system functions. PHYSICAL IDs are assigned at initialization time and provide a system-wide unique identifier for each agent. LOGICAL IDs are assigned to agents or groups of agents to implement and/or facilitate certain higher system functions such as fault tolerant agents and one of the group protocols.

The Internal Destination is used to define the action of the IAC and provide modifiers for that action. There are two types of action that are defined: the IAC message and Register/Command access is used to read or write internal registers in the agent or force an action such as initialization of an agent.

1. IAC Messages:

IAC messages are from 1 to 4 words in length and are sent only by a Write Word(s) or Write Partial Word(s) where the Write Partial can specify only words. As before, a single cycle packet must be followed by a null cycle to allow the AP Bus system time to respond. The packet has a priority associated with it that is indicated in the Internal Destination Field. This is the packet priority and will be used to determine the acceptance or refusal of the message. Any Reply (other than a Read Reply) is a legitimate Reply.

Two bytes of the data field are fixed. They are called the Message-type field and the Message-priority field. Both are one byte in length and will be used by the higher level IAC protocols. The Message-priority and the packet priority are not required to be the same.

The priority field in the Internal Destination Field (packet priority) is 5 bits long and can range in value from 0 to 31. The packet priority is used by the addressed agent to determine if it will accept the message. A priority of 0 is the lowest and a priority of 31 is the highest. Priority 31 is considered a non-maskable interrupt (NMI) and must be accepted by the addressed agent.

2. Command/Register Access:

Command/Register accesses are Write Word(s) and Read Word(s) Requests. They may access only one register/command per transaction. The Register and Commands are typically one word long but there is no restriction. The Request must match the register/command size for the Request to be valid. All Replies are valid for Command and Register transactions.

F. Transaction Protocol:

The separation of a bus transaction into a Request Packet and a Reply Packet provides for multiple transactions on the bus. Each packet occupies only the number of sequential bus cycles necessary to transmit the data it carries. There will typically be a delay between the Request and Reply which would normally be unused bus bandwidth. The AP Bus uses these cycles to pipeline multiple transactions on the bus.

The AP Bus utilizes a pipeline of requests, allowing several transactions to be in progress at any one time. Each transaction occupies one slot in the pipeline. As a Reply is made to a Request, the transaction is removed from its slot which is then available for another transaction.

Multiple agents may request and use the bus in an ordered multiplexed fashion. They gain the use of the bus through arbitration. Arbitration orders and prioritizes(?) the agents' requests for the bus and places them in a Grant Queue. The agent at the top of the Grant Queue will take the next empty time slot of the AP Bus. Figure 6 illustrates the relationship between arbitration, sequencing of Request and Reply Packets, and ordering of Replies.

1. Arbitration:

The BXU's will arbitrate between themselves on behalf of their attached processors, to obtain access to the AP Bus. Memory Arrays do not initiate requests; therefore, MCU's don't arbitrate. The arbitration algorithm chosen for the AP Bus supports distribution of the implementation across the BXU's. The algorithm guarantees that no single BXU will be locked out from accessing the bus. Arbitration itself is based on a 6-bit arbitration identifier, or ARBID (number stored in a register) see Figure 7. Each BXU has its own unique ARBID that is used to determine its priority in each arbitration cycle or Time Slice. All BXU's arbitrating in a Time Slice will be placed in the Grant Queue, GQ. The top of the Grant queue GQ is the next agent to be placed on the bus. Note that BXU's acting as memory array controllers only do not make Requests and therefore do not arbitrate.

A Time Slice is a period of bus cycle in which grants are made for BXU's to access the bus. A Time Slice begins when the previous Time Slice has comleted and there are BXU's that desired to use the bus. All BXU's ready to arbitrate in the first cycle of a Time Slice will be included in that Time Slice. Any BXU that determines it needs the bus while a Time Slice is in progress must wait until the next Time Slice to arbitrate. The length of the Time Slice will be determined by the number of agents involved, the ARBIDs of the agents and the available depth of the Grant Queue GQ.

The four ARB (ARB3...∅) lines are used in implementing the arbitration algorithm. They are wire-or signals driven by all active agents and monitored by all agents. They will be used to signal the beginning of a Time Slice and determine the next BXU to be placed in the Grant Queue. Each Time Slice will have from 1 to 32 Time Steps. Within each Time Step up to three grants can be made.

The ARBID, Arbitration Identification Number, as shown in Figure 6, has two fields that determine the grant order. The COUNT FIELD CF determines the number of Time Steps required by the agent before its Time Step. The DRIVE FIELD DF determines the agent's priority within the Time Steps TS. Figure 5 further illustrates the relationship of the DRIVE FIELD to the ARB lines.

The COUNT FIELD CF in an ARBID is from 0 to 31 in value. A COUNT FIELD CF of 0 indicates that the BXU will arbitrate in the first Time Step of a Time Slice while a value of 31 indicates that it will arbitrate in the thirty-second Time Step. The DRIVE FIELD DF is used for parallel arbitration within a Time Step. Agents having the same COUNT value in their ARBID will arbitrate in the same Time Step. The DRIVE FIELD DF encoding determines which of the three parallel arbitration lines, ARBO through ARB2, the agent will assert during this Time Step. The agent asserting ARBO will be placed in the Grant Queue GQ first, followed by the one asserting ARB1, then the one asserting ARB2.

ARB3 is a logical "OR" signal between the BXU's and when asserted by one or more units pulling the voltage on the line to a low voltage, will indicate if there are any more Time Steps required to complete the Time Slice. ARB3 will be asserted by all agents in the Time Slice that require additional Time Steps. The COUNT FIELD in an agent's ARBID determines how many Time Steps the agent will drive ARB3 and where it will assert its parallel arbitration line. For example, if an agent's COUNT FIELD is 3, then it will assert ARB3 for three Time Steps and then assert its parallel arbitration line of the fourth Time Step. This can be determined by counting back the count field number to zero. The Time Slice will be complete if there is no other agent asserting ARB3 in that fourth Time Step.

A Time Step's duration is determined by the number of parallel arbitration lines asserted during the first bus cycle of the Time Step and the available depth of the Grant Queue GQ. Each agent asserting one of the three parallel arbitration lines will be placed sequentially into the Grant Queue. If all three lines are asserted, then the agent asserting ARBO will be placed in the Grant Queue during the first cycle of the Time Step. The agent asserting ARB1 will be assigned on the second cycle and the one asserting ARB2 on the third. The Time Step is complete when all the agents arbitrating in parallel are assigned to the Grant Queue. If only two lines are asserted, the agent asserting the higher priority will be placed in the Grant Queue during the first cycle of the Time Step and if the Grant Queue is still not full, the remaining agent asserting an arbitration line will be placed in the Grant Queue on the subsequent Time Step. If only one line is asserted, the agent asserting that line is placed in the Grant Queue on the first Time Step.

In both single agent and multiple agent embodiments, Grant Queue GQ contains eight entries. Arbitration will be suspended whenever there are eight entries in the queue. The state of the ARB-lines will freeze until the queue is popped and there is an open entry. A Time Step will be stretched beyond the normal maximum of three cycles by a full Grant Queue.

Figure 7 illustrates the interaction of the ARB-lines to define the Time Steps and the Time Slice. The Time Slice is begun when the four agents A, B, C, D that desire to arbitrate, assert the ARB-lines based on their ARBIDs. The two agents, A, B with a COUNT FIELD of ∅ assert their associated parallel arbitration lines. Agent A asserts ARBØ and B asserts ARB1. Agents C and D both assert ARB3 to indicate that more Time Steps will be required.

The Time Slice begins in the bus cycle numbered 2 and agent A is assigned to the Grant Queue at the end of that cycle. Cycle 3 shows the Grant Queue increased by 1 to 6 and A no longer asserting ARBØ. B continues to assert ARB1 during cycle 3 and is assigned to the Grant Queue at the end of the cycle thus increasing the queue depth to 7 and completing the Time Step. The next Time Step has no agents arbitrating and therefore takes only one clock cycle.

In cycle 5 the final Time Step is begun and therefore ARB3 is no longer asserted. Agent C is assigned to the Grant Queue thus increasing it to its maximum value of 8. This halts arbitration until cycle 8 where an agent gets on the bus and is removed from the Grant Queue. The Time Step and Time Slice are now completed as agent D is assigned to the Grant Queue.

2. Bus Sequencing:

The section on arbitration defines the process for ordering agents' access to communicate via request packets over the AP-Bus. Bus Sequencing defines how Request and Reply packets are interspersed on the AP Bus.

Bus sequencing determines the use of the next bus cycle. This determination is based on the state of the reply deferral, RPY, DEF ARB and SPEC lines sampled at the end of the cycle just completed as well as the present state of the Grant and Pipeline Queues. The term "next available bus cycle" used in the following rules refers to the cycle following the packet presently being transmitted on the bus. If there is not a packet on the bus then the cycle in which the state is being examined is considered the next available cycle.

The following rules govern Bus Sequencing:

1. If there is an agent identified in the Grant Queue and the Pipeline Queue is not full (less than three transactions), then a Request must take the next available bus cycle.

2. If the Pipeline Queue is full (three transactions in progress) then a Reply may take the next available bus cycle.

3. If the Grant Queue is empty, but the Pipeline Queue is not empty, then a Reply may take the next available bus cycle.

The first rule provides the normal ordering when the Pipeline Queue is not full. Giving Requests precedence over Replies allows the pipeline to be kept as full as possible. Keeping the pipeline full maximizes the use of the available bus bandwidth by having as many transactions in progress as possible. This also maximizes the use of available resources since there can be more than one memory controller (MCU or BXU) attached to the bus. If there are transactions to more than one memory controller, then they will be overlapped.

The second rule declares that once the pipeline is full, then only Replies can be placed on the bus. A Reply reduces the Pipeline Queue depth which would then place the first rule into effect again. If the Grant Queue is not empty, then a Request would go next.

The third rule allows the Pipeline Queue to be emptied when there are no outstanding Requests in the Grant Queue.

It is important to notice that the next reply is not forced onto the bus at any time. Just because the RPY DEF signal lines are not asserted, does not mean that the reply is ready. RPY DEF is only used when the agent knows that its reply will be much slower than the normally expected access time. Thus, the bus sequencing rules define time windows when a reply can go, but it does not force a certain access time on the replying agent.

3. AP-Reply Ordering:

The preferred embodiments of the bus can have up to three agents in the pipeline waiting to transmit Replies. Each of these agents occupies a slot in the Pipeline Queue PQ. Each Request on the bus indicates the agent required to Reply and the Replying Agent is identified in a slot in the Pipeline Queue. Slot 1 is the top of the queue and will be occupied by the agent that has been in the queue the longest. Slot 2 contains the second longest while slot 3 contains the most recent agent to be placed in the queue. The ordering of Replies to these transactions is controlled by the Request sequence (as indicated in the Pipeline Queue PQ); however, the Reply deferral mechanism can under certain circumstances indicated in the rules following, alter the normal sequence of Replies.

The following rules govern Reply Ordering:

1. Replies will normally be returned in the order in which the Requests were made. The next Reply would, therefore, be to the transaction in slot 1 of the Pipeline Queue.

2. The replying agent assigned to slot 1 may defer its own slot in the Pipeline Queue until later. This permits a transaction in slot 2 of the Pipeline Queue to be completed before the one ahead of it.

The Replies will normally be returned in the same order that the Requests were placed in the Pipeline Queue. Thus, the next Reply on the bus will be for the agent identified in slot 1. The Reply for slot 1 will complete that transaction and remove it from the queue. Then all indicated in the Pipeline Queue will move up one slot. The agent in slot 2 will move the slot 1 and the agent in slot 3 to slot 2. The Reply to the new slot 1 will be the next reply normally placed on the bus.

The Reply ordering is modified by the Reply Deferral (RPYDEF) signal on the Reply Deferral control line. An agent replying to a Request may defer its slot in the Pipeline Queue by asserting the RPYDEF signal. The RPYDEF line may only be asserted by the replying agent that owns the reply in slot 1. If RPYDEF is asserted, the reply currently associated with slot 1 will be placed at the back of the pipeline queue.

Reply deferral also inhibits the timeout function for the request being deferred. When an agent requires an extended time to make its Reply, it must assert its RPYDEF line to stop the timeout. If the BUS SWITCHING enable control bit is set in the BXU's and MCU's, then RPYDEF only stops the timeout and does not cause reordering of the replies.

G. Bus Details:

As can be seen from Figure 1, the individual agents A, B (VLSI-chips in the preferred embodiment) are connected via associated BXU's or MCU's to the AP Bus. Each BXU and MCU has integrated bus control logic BCL connecting these units to the lines of the bus. The bus control logic BLC with its control part CL and its intermediate register ystem CM for address and data serves to interface a high speed local processor bus to the very high speed AP Bus.

As described, the lines ARBO-3 are used for arbitration, the lines RPY DEF for reply ordering betwen the agents and additionally lines SPECØ-5 for sequencing. The Arbid-Register AR with its parts CF (count field) and DF (drive field) specifies numbers for time priority (count field) and priority (drive field) between agents which arbitrate in the same time step (Figures 7 and 8) and is connected to the lines ARB Ø-3. If the priority has been decided according to the arbitration logic, a gate logic GL puts the winning agent into the register Grant Queue GQ. The order of the Grant Queue is stored by definite signals in each agent. The same holds for the register Pipeline Queue PG which determines the reply order according to the logic indicated in Figure 8.

It will now be understood that there has been disclosed an Advanced Processor Bus System for a VLSI data processor which has a high effective data bandwidth because of the 32 parallel address-and-data lines and the associated parallel specification and control lines, many which are multiplexed to perform different functions in different times. In addition to the apparatus that provides a high effective data bandwidth, also disclosed are pipelining and arbitration procedures and packet protocols that provide for a high utilization of the effective bandwidth. Moreover, the disclosed AP Bus system apparatus and procedures are flexible and expandable to multi-AP Bus systems and to multi-processor AP Bus systems to suit the system needs because of the modularity of the apparatus and the procedures that make up the AP Bus system. As will be evident from the foregoing description, certain aspects of the invention are not limited to the particular details of the examples illustrated, and it is therefore contemplated that other modifications or applications will occur to those skilled in the art. It is accordingly intended that the claims shall cover all such modifications and applications as do not depart from the true spirit and script of the invention.

Claims

1. An Advanced Processor Bus system for communicating data words among at least one processor connecting to the AP Bus by at least one bus interface means such as a Bus Expander Unit; and at least one memory array connecting to the AP Bus by at least memory control mans, such as a Memory Control Unit, wherein said bus interface means connected to respective processors issue data requests request the transmission of data upon the AP Bus for reception into a register of each said bus interface means making such request; and said bus interface means and said memory control means issue data replies communicating over the AP Bus in response to the data requests by transmission of the requested data, memory control means issue data replies communicating over the AP Bus in response to the data requests by transmission of the requested data, said AP Bus system comprising:
message generator means for generating messages in the form of packets of information for transmission on said Bus system, said messages being divided into message types including control message packets, request message packets, and reply message packets, each packet comprising at least one bus transmission slot issued by said message generator means sequentially and contiguously, each bus transmission slot in a packet being capable of including operation specification code, address, data, control, and parity-check bits;
a pipeline queue;
message controller means connected to said message generator means and to said pipeline queue for controlling said request message packets, said control message packets and said reply message packets such that a predetermined number of said request message packets and control message packets requiring reply message packets may be entered into said pipeline queue at any one time;
monitor means connected to said message controller means and to said Bus system for monitoring said request message packets and control message packets, and said reply message packets communicated on said bus by said message generator, preventing request message packet replies in excess of said predetermined number from being communicated on said bus until a reply message packet is communicated on said bus to thereby make available a slot in the pipeline queue;
control signal connections as a part of said AP Bus operative in parallel with the data portion thereof for providing a coded signal indicating a particular message type communicated on said bus by said message generator; and
interface logic means connected to said message generator means and to said control signal lines for communicating over said AP Bus system the type of message packet generated;
said message generator means including means for inserting a particular reply message packet cor-

responding to a particular request message packet in said pipeline queue at a position in said pipeline queue corresponding to the request message packet that is associated with said particular reply message packet.

2. An Advanced Processor Bus system for use in a data processing system using VLSI Advanced Processor components, having AP Bus interface units for processors as AP nodes and memory interface units as AP Bus nodes; said AP Bus connecting to and communicating among each of said nodes, comprising:
recognizing means connected to said Advanced Processor Bus for recognizing signals on said AP Bus representative of a request to one of said nodes;
means connected to said AP Bus, responsive to said recognizing means, for generating on said Bus, signals representative of a first and a second write-request packets, and a first and second read request packets, said packets comprising a number of bus transmission slots, issued sequentially and contiguously, each slot in a packet including:
operation-specification bits;
address bits;
data bits; and
control bits;
said first write-request packet including an operation specification code specifying a write function to be performed and a memory location where the information bits are written to, and said second write-request packet including an operation specification code specifying a function to be performed and a memory mapped location where the information bits are to be written to;
said first read request packet including an operation specification code specifying a read function to be performed and a memory location where the information bits are read from, and said second read request packet including an operation specification code specifying a function to be performed and a memory mapped location where the information bits are read from;
said memory mapped locations corresponding to inter-agent communications specifying an internal destination code of an agent and the functions desired of the specified agent;
receiving means connected to said AP Bus for receiving a reply message packet on said AP Bus system, said reply message packet including a number of bus transmission slots, received sequentially and contiguously, each slot in said reply packet including operation specification code, data bits and control bits;
said reply message packet containing an operation specification code and if requested data;
message control means connected to said generating means and said receiving means said message control means including as a part of said message control means:

means for queuing data in said write request packets,
means responsive to said control bits in said read packets, and write packets for detecting a beginning and an end of each packet,
means for decoding each said operation specification code to indicate the operation specified,
means for establishing a pipeline for said request message packets, and said reply message packets generated on said AP Bus by said message generator,
monitor means connected to said message control means and to said AP Bus system, for monitoring signals on said AP Bus system representative of said request packets, and reply packets on said bus, such that a request packet is prevented from entering said pipeline until a reply packet is received to make available a slot in said pipline, whereby request packets increase the length of said pipeline up to a predetermined number of request packets in said pipeline, and reply packets decrease the length of said pipeline; and
means connected to said recognizing means, said generating means, said message control means and said receiving means, for arbitrating said signals on said Bus system representative of requests going onto said Bus system and said signals on said Bus system representative of replies received on said Bus, to enable said message control means to insert a reply packet associated with a particular request packet in the pipeline position of said recognized request packet.

3. The combination as set forth in claim 1, wherein said monitor means includes:
reply monitor means including means for maintaining a queue of reply packets in the same order as the associated request packets;
means for informing a replying unit when it is time to reply; and
means for informing a requesting unit when an associated reply packet is to be received.

4. The combination as set forth in claim 2, wherein said monitor means includes:
reply monitor means including means for maintaining a queue of reply packets in the same order as

the associated request packets, means for informing a replying agent when it is time to reply and means for informing a requesting agent when an associated reply packet is to be received.

5. The combination as set forth in claim 4, further comprising:
re-ordering means connected to said message control means, said generating means and said receiving means to enable said message control means to change the pipeline position which the reply packet associated with a particular request packet is inserted on the Advanced Processor Bus.

6. The combination as set forth in claim 1, further comprising:
means connected to said monitor means and said message controller means, for arbitrating said signals on said Advanced Processor Bus representative of requests from said Advanced Processor Bus, and signals on said Advanced Processor Bus representative of replies received from said Advanced Processor Bus, to thereby enable said message controller means to insert a reply packet associated with a particular request packet in the pipeline position of said request packet.

7. The combination set forth in claim 6, further comprising re-ordering means connected to said monitor means and said pipeline queue means for changing the order that the reply message packets are communicated onto said bus system by deferring the use of the next available slot in the pipeline to a reply of lower priority within the pipeline queue, changing the order of the reply message packets in the pipeline.

8. The combination as set forth in claim 7, wherein said means for generating packets includes:
means for generating on said AP Bus a write-request packet and a read-request packet,
said write-request packet including an operation specification code specifying an operation desired, a memory mapped address, or an address specifying a physical memory location and write data,
said read-request packet including an operating specification code specifying an operation desired, a memory mapped address, or an address specifying a physical memory location; and
wherein said message controller means further comprises:
means for queuing data in said write request packet;
means responsive to said control bits in said read and write packets for detecting the start and end of a packet; and,
means for decoding said operation specification code to provide an indication of a function.

9. The combination as set forth in claim 7, further comprising:
means connected to said monitor means and said message controller means, for arbitrating said requests onto said bus and said replies received from said bus, to thereby enable said message controller means to insert a reply to a particular request in the pipeline queue at a position in said pipeline queue corresponding to the request message packet that is associated with said particular reply message packet.

10. The combination as set forth in claim 7, wherein said means for generating packets includes:
means for generating on said AP Bus a write-request packet and a read-request packet,
said write-request packet including an operation specification code specifying a function desired, a memory mapped address or address specifying a physical memory location, and write data,
said read-request packet including an operation specification code specifying a function desired, and an address specifying a physical memory location; and
wherein said message controller means further comprises:
means for queuing data in said write request packet;
means responsive to said control bits in said read and write packets for detecting the start and end of a packet; and
means for decoding said operation specification code to provide an indication of a function.

11. The combination as set forth in claim 7, wherein said means for generating packets includes:
means for generating on said bus a write-request packet and a read-request packet,
said write-request packet including an operation specification code specifying an operation desired, a memory mapped address or an address specifying a physical memory location; and write data,
said read-request packet including an operation specification code specifying a function desired, a memory mapped address, or an address specifying a physical memory location; and
wherein said message controller means further comprises:
means for queuing data in said write request packet;
means responsive to said control bits in said read and write packets for detecting the start and end of a packet; and
means for decoding said operation specification code to provide an indication of an operation.

12. The combination as set forth in claim 5, further comprising:
means connected to said monitor means and said message controller means, for arbitrating said requests onto said AP Bus and said replies received from said AP Bus, to thereby enable said message

controller means to insert a reply to a particular request in the pipeline queue at a position in said pipeline queue corresponding to the request message packet that is associated with said particular reply message packet.

13. The combination as set forth in claim 5, wherein said means for generating packets includes:
means for generating on said AP Bus a write-request packet and a read-request packet,
said write-request packet including an operation specification code specifying an operation desired, a memory mapped address, or an address specifying a physical memory location and write data,
said read-request packet including an operation specification code specifying an operation desired, a memory mapped address, or an address specifying a physical memory location; and
wherein said message controller means further comprises:
means for queuing data in said write request packet;
means responsive to said control bits in said read and write packets for detecting the start and end of a packet; and
means for decoding said operation specification code to provide an indication of an operation.

14. The combination as set forth in claim 5, wherein said means for generating packets includes:
means for generating on said bus a write-request packet and a read-request packet,
said write-request packet including an operation specification code specifying an operation desired, a memory mapped address, or an address specifying a physical memory location; and
pipeline queue means for changing the order that the reply message packets are communicated onto said bus system by deferring the use of the next available slot in the pipline to a reply of lower priority within the pipeline queue, changing the order of the reply message packets in the pipeline.

15. The combination as set forth in claim 1, further comprising: means for initializing the Advanced Processor Bus system, including a data portion, for identifying each major unit of said AP Bus system and providing to each said major unit instructions as to processes to be performed.

16. The combination as set forth in claim 2, further comprising: means for initializing the Advanced Processor Bus system, including a data portion, for identifying each major unit of the system and providing to each said major unit instructions as to special processes to be performed.

17. The combination as set forth in claim 1, further comprising: means for aiding diagnostics connected to said message generator means, said interface logic means and said control signal means whereby a bus command can be given stopping the information on the AP Bus in place for one bus cycle and including the capacity to repetitively give said information stopping command.

18. The combination as set forth in claim 2, comprising: means for aiding diagnostics connected to said message generator means, said interface logic means and said control signal means whereby a bus command can be given stopping the information on the AP Bus in place for one bus cycle and including the capability to repetitvely give said information stopping command.

88P7462E

FIG. 1

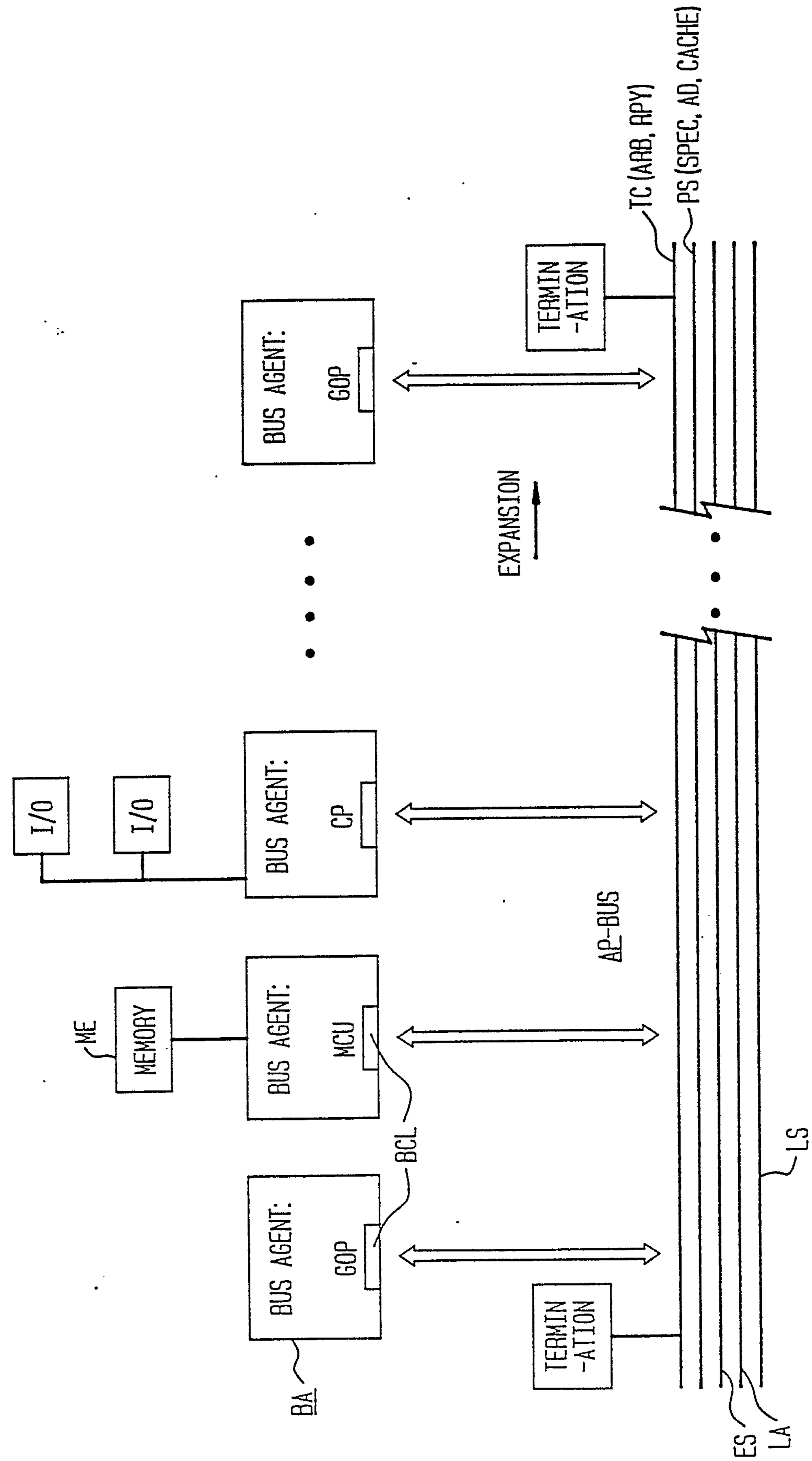


FIG. 2

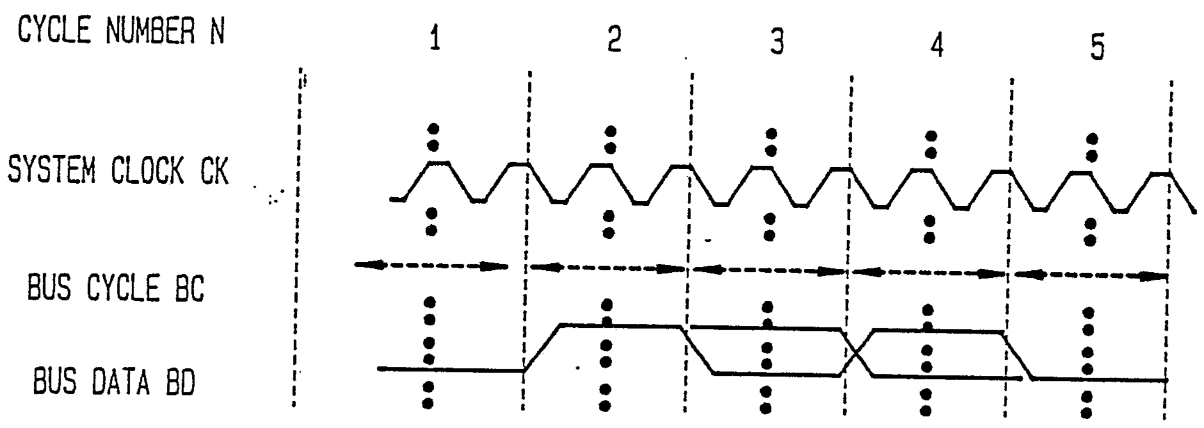
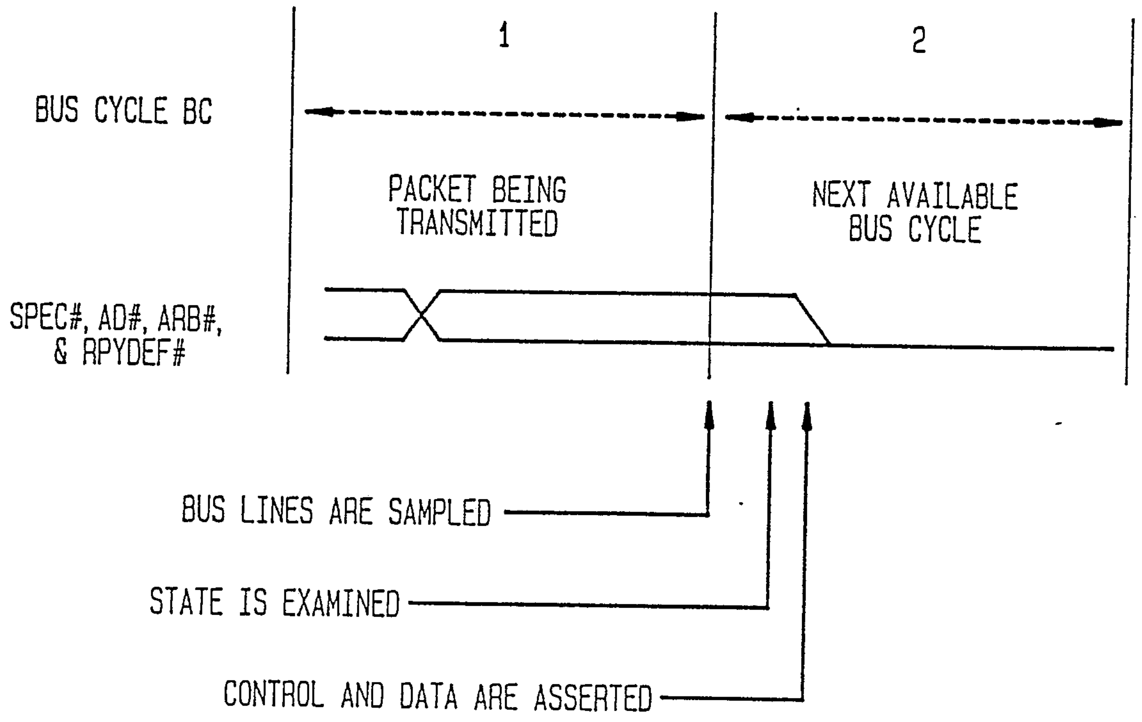

CYCLE NUMBER N
1
2
3
4
5
SYSTEM CLOCK CK
BUS CYCLE BC
BUS DATA BD

FIG. 3


1
2
BUS CYCLE BC
PACKET BEING TRANSMITTED
NEXT AVAILABLE BUS CYCLE
SPEC#, AD#, ARB#, & RPYDEF#
BUS LINES ARE SAMPLED
STATE IS EXAMINED
CONTROL AND DATA ARE ASSERTED

88P7462E

FIG. 4

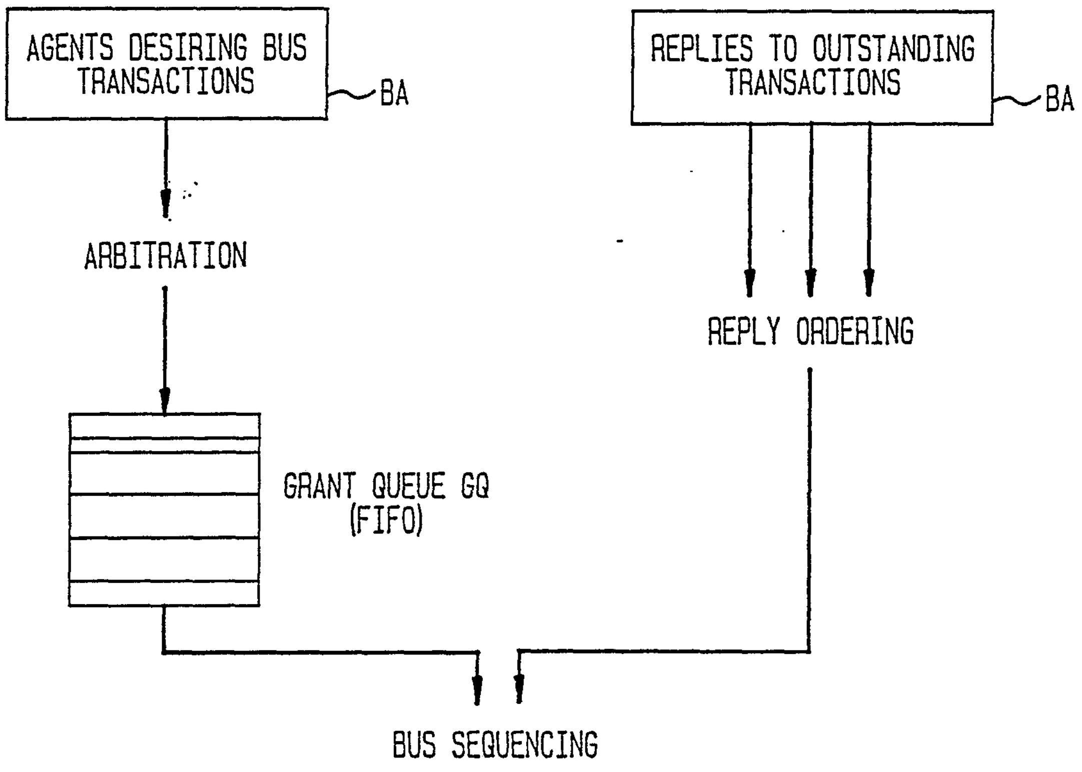
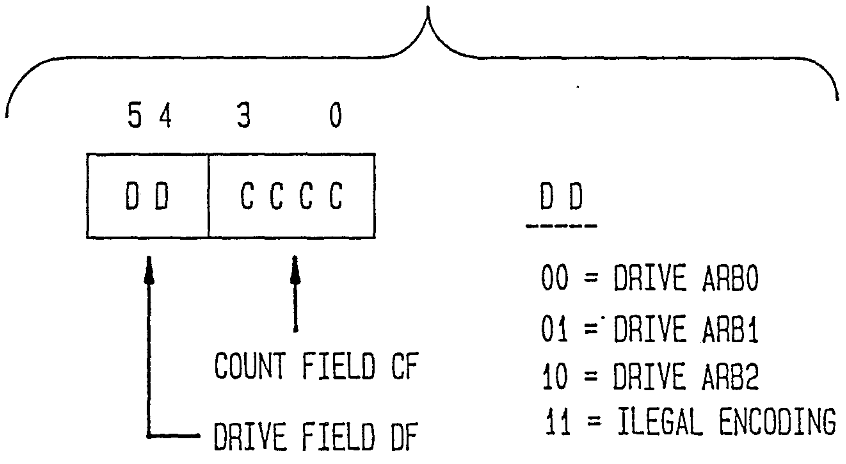
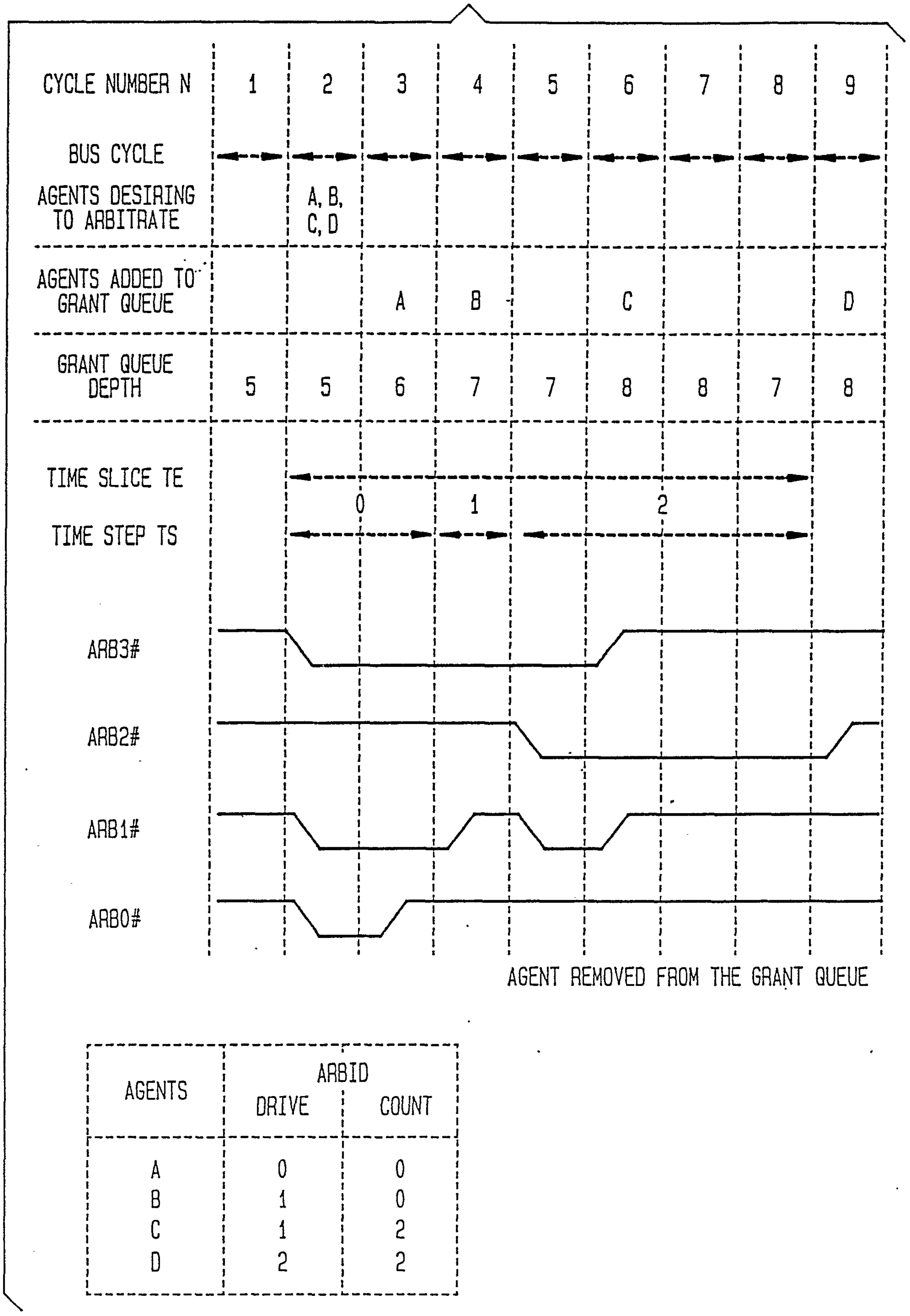

AGENTS DESIRING BUS TRANSACTIONS
BA
REPLIES TO OUTSTANDING TRANSACTIONS
BA
ARBITRATION
REPLY ORDERING
GRANT QUEUE GQ (FIFO)
BUS SEQUENCING

FIG. 5



88P7462E

FIG. 6



AGENTS	ARBID DRIVE	ARBID COUNT
A	0	0
B	1	0
C	1	2
D	2	2

88P7462E

FIG. 7

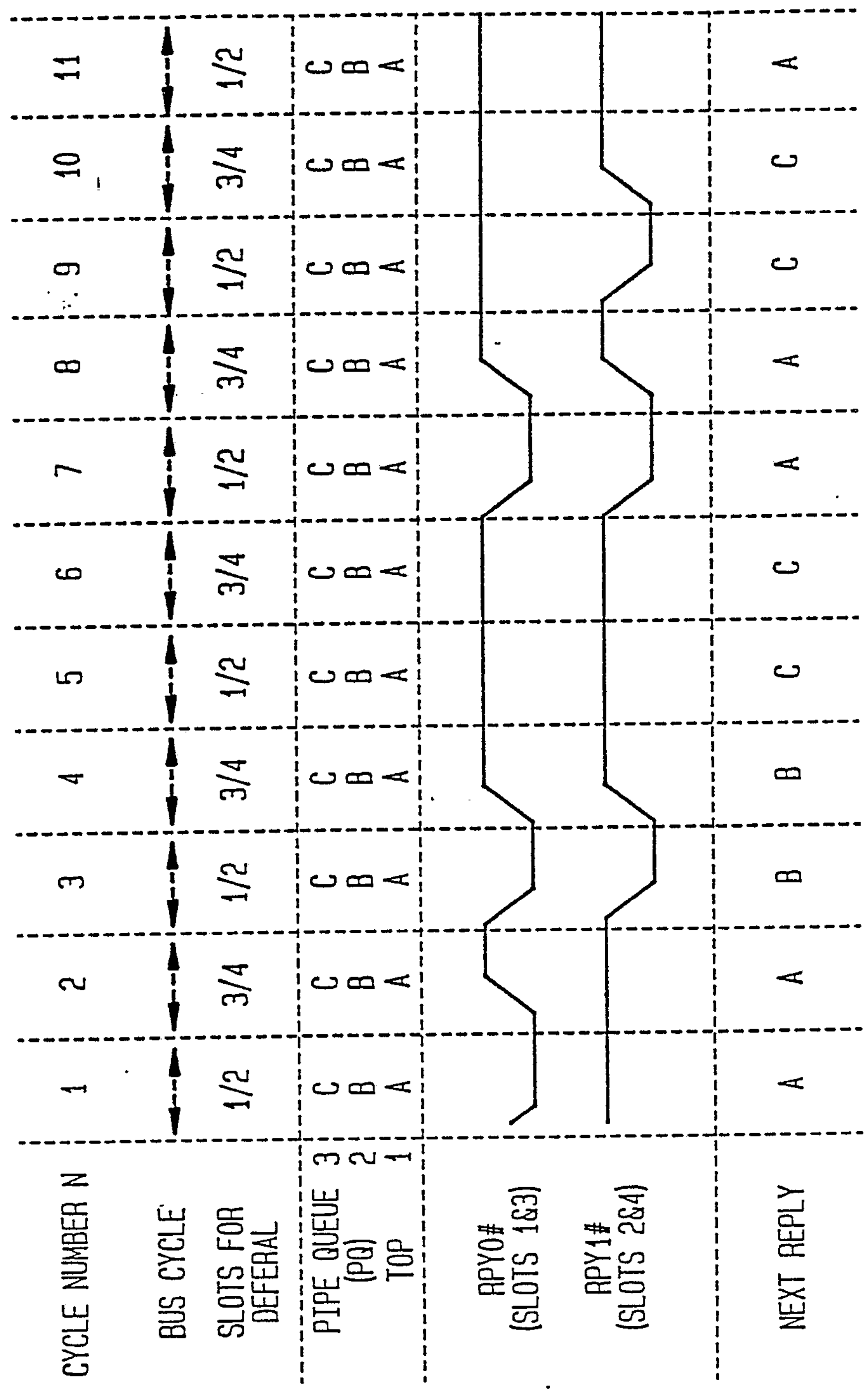


FIG. 8

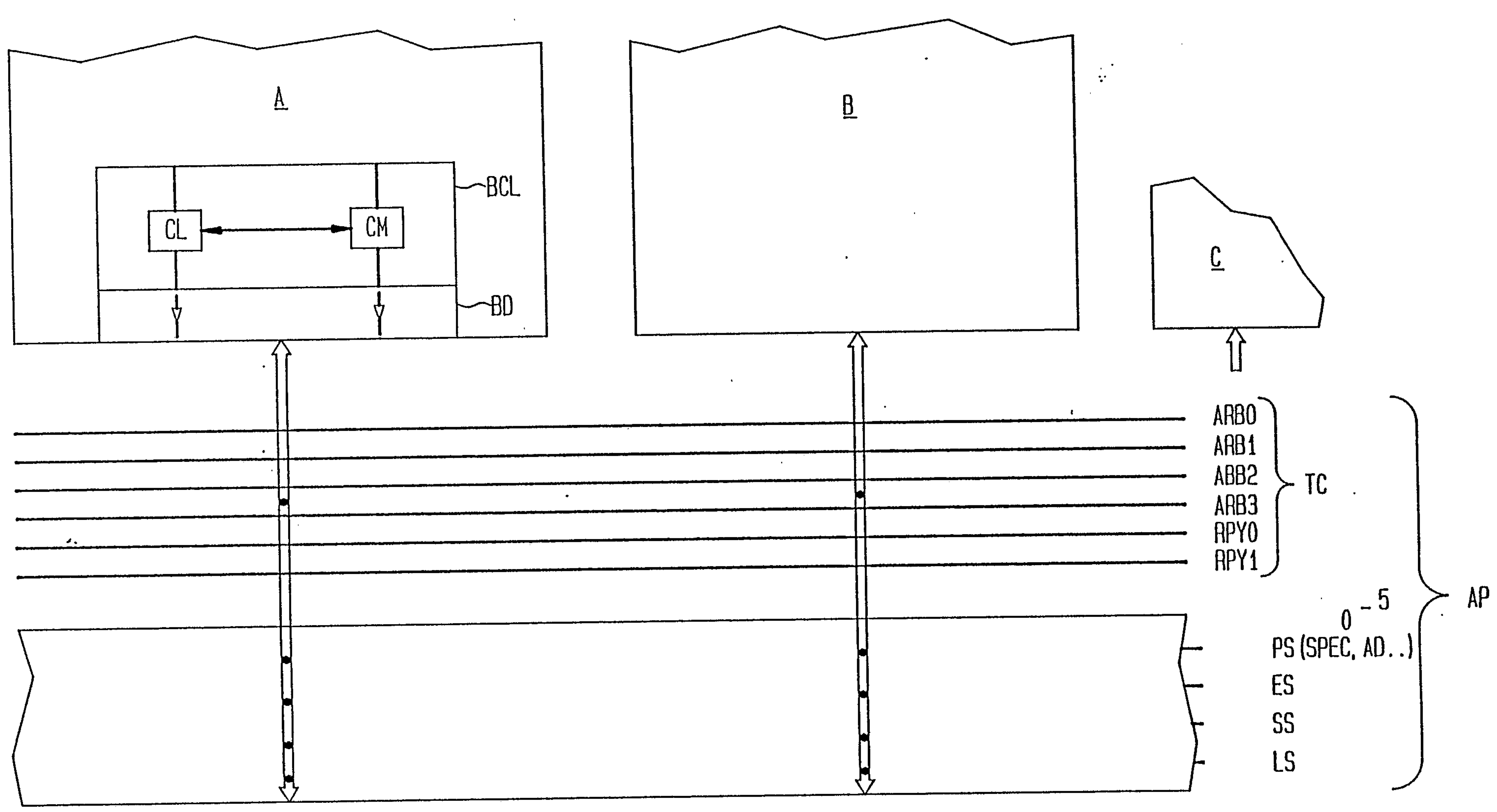
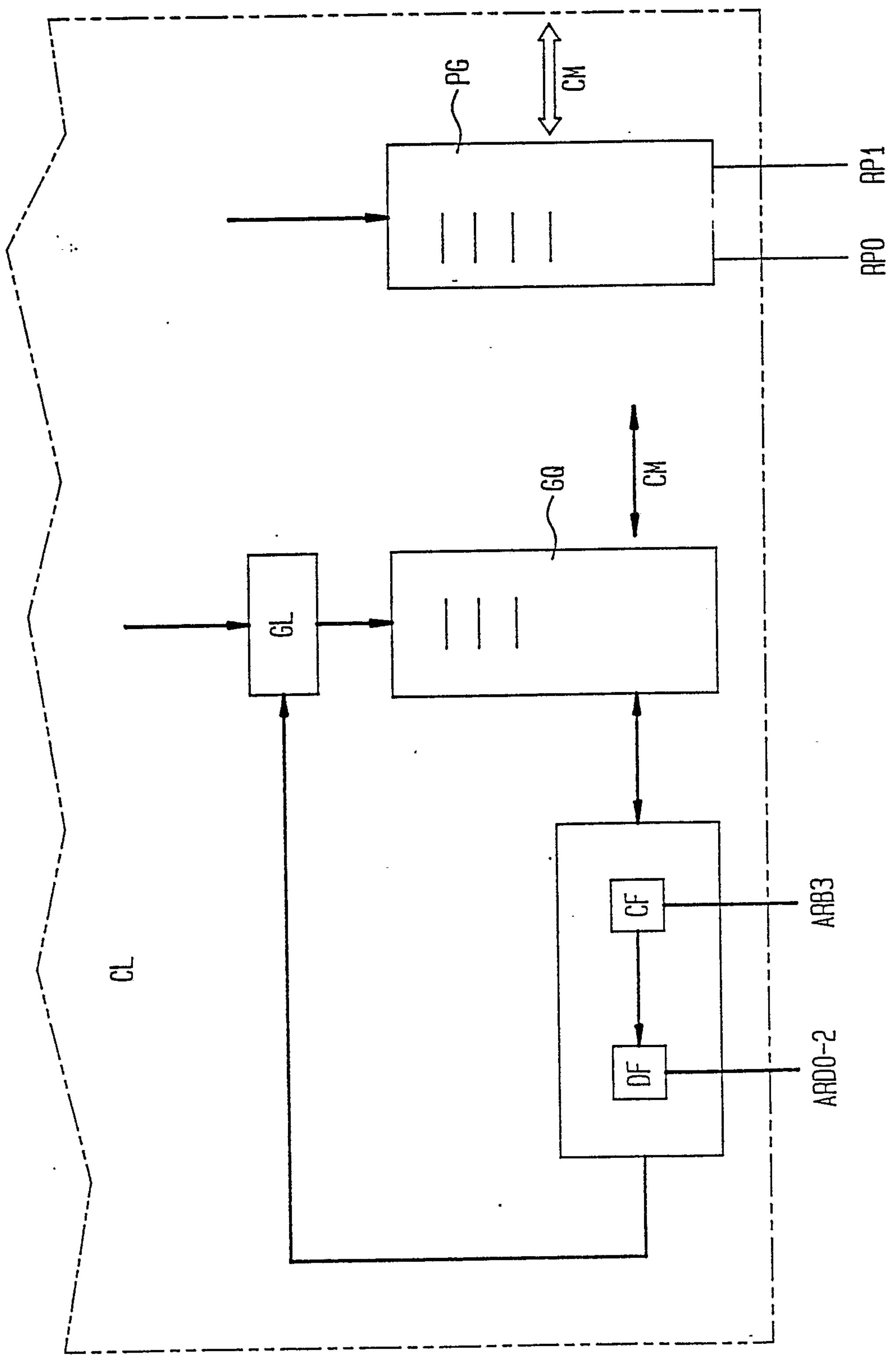

A
BCL
CL
CM
BD
B
C
ARB0
ARB1
ABB2
ARB3
RPY0
RPY1
TC
0 - 5
PS (SPEC, AD..)
ES
SS
LS
AP

FIG. 9

A


PG
CM
RP1
RP0
GQ
CM
GL
CF
DF
ARB3
ARD0-2
CL